Express Mail No. EL414495568US
Docket No. BO9-99-030
Firm No. 0036.0044

METHOD, SYSTEM, AND PROGRAM FOR PROCESSING A JOB IN AN EVENT DRIVEN WORKFLOW ENVIRONMENT

Inventors: Leonard C. Lahey, Robert C. Nielsen, Dwight R. Palmer, Adam A. Swartz

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method, system and program for processing a job in a workflow environment using event driven programming to route the job through the workflow environment.

2. Description of the Related Art

A workflow management system provides a defined series of tasks within an organization to produce a final outcome. Sophisticated workgroup computing applications define different work flows for different types of jobs. At each stage in the workflow, one individual process or application is responsible for a specific task. Once the task is complete, the workflow program ensures that the individuals or processes responsible for the next task are notified and receive the data needed to execute the next stage in the workflow process.

Some workflow management systems provide complex schemes involving rules and graphs to determine how to route a job through a system. Others poll a data set to determine if any new statuses or flags are set indicating a need to proceed to the next step in the workflow.

However, there is a need in the art for an improved method and system for managing the flow of work in a computing system.

SUMMARY OF THE PREFERRED EMBODIMENTS

To overcome the limitations in the prior art described above, preferred embodiments disclose a method, system, and program for processing a job in a workflow environment. A signal is generated when status for the job is changed from

a first status to a second status. A work process associated with the second status is notified that one job had its status changed to the second status in response to the signal. The work process processes the job that had its status changed from the first status to the second status and modifies the status of the job after completing the processing of the job.

In further embodiments, the job status is maintained in a database table including information on the job. The work process maintains a connection with the database that enables communication with the database table. The work process modifies the status of the job after completing processing by updating the status of the job to an output status associated with another work process. Updating the status with the output status generates the signal indicating a change in status.

In still further embodiments, there are multiple work processes each associated with an status and each enabled to update the job status with the output status after completing the processing of the job. The output status for one worker is the status associated with one other worker. In such case, the job proceeds according to the order in which the work processes update the jobs with their output status.

In even further embodiments, the work process spawns a work thread to process one job in the database table having the status associated with the work process. The work process is capable of spawning multiple work threads to process different jobs having the status associated with the work process.

Preferred embodiments provide an event driven workflow system. A job enters the workflow system by having its status set to the first possible status value. Setting or updating status triggers an event which signals a worker process associated with the status to process the job. After the job is processed, the worker process updates the status to an output status which triggers another event signal to another worker process associated with the output status to proceed with processing the job. In this way, a change in the status of the job drives the workflow environment to provide a just-in-time type system for processing jobs using database technology.

Preferred embodiments are implemented using push technology to notify worker processes to process the job when the job is ready for such processing.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a computing environment in which preferred embodiments of the present invention are implemented;

FIG. 2 illustrates a job status table used with preferred embodiments of the present invention;

FIG. 3 illustrates logic implemented in the database and user defined function (UDF) to manage workflow of jobs in accordance with preferred embodiments of the present invention;

FIG. 4 illustrates logic implemented in a worker process to process a job in accordance with preferred embodiments of the present invention; and

FIG. 5 illustrates logic implemented in a worker thread to process a job in accordance with preferred embodiments of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments of the present invention. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

Computing Environment

FIG. 1 illustrates a computing environment in which preferred embodiments are implemented. A computing system 2 may comprise one or more workstations,

where the server 4 and database 6 are implemented in one workstation and the workers 8a, b, c, d are implemented in one or more computer workstations, which may include the server 4 or any other distributed computing device that forms the computing system 2. Different devices in the computing system 2 could communicate using any communication technology and protocol known in the art, such as a telephone line, dedicated cable or network line, etc, using any protocol known in the art including TCP/IP network (e.g., an Intranet, the Internet), LAN, Ethernet, WAN, Token Ring, etc.

The server 4 and any other clients in the computing system 2 would include an operating system, such as MICROSOFT WINDOWS 98 and WINDOWS NT, AIX, OS/390, OS/2, OS/400,** and may be comprised of any suitable server and client architecture known in the art. The server 4 includes a database program 6. The database program 6 may be comprised of any database program known in the art, such as DB2, Oracle Corporation's ORACLE 8, Microsoft SQL Server,** etc. The database program 6 is used to manage a job status table 10 and control the execution of a user defined function (UDF) 12 from a call within the database 6. The UDF 12 may be written in JAVA.** Further details of the architecture and operation of a database program are described in the IBM publications "DB2 for OS/390: Administration Guide, Version 5" IBM document no. SC26-8957-01 (Copyright IBM. Corp., June, 1997) and "A Complete Guide to DB2Universal Database," by Don Chamberlin (1998), which publications are incorporated herein by reference in its entirety.

FIG. 2 illustrates an example of an implementation of the job status table 10, including: a Job ID column that is a job identifier, which may be the primary key for the job record in the job status table 10; a current status column including the current status of the job, e.g., ready, print, format, etc., which may describe the process currently being applied to the job; a time stamp column indicating the time the current status was applied; a priority of the job; and other job information. The job status

table 10 may be a separate table that is related to another table that includes further information on each job identified in the job status table 10 or may comprise two columns of a larger table that includes further information on each job. Each job identified in the job status table 10 comprises an entity on which work is performed under computer operation such as processing data, generating output materials, forwarding data to another location for further processing, printing, working on a material or device, etc.

The workers 8a, b, c, d comprise separate application programs that perform the different work steps in the system 2. Each worker 8a, b, c, d may execute as a separate daemon process that maintains a dedicated connection 16a, b, c, d to the database 8, in a manner known in the art, to communicate with the database 6. In such case, the worker 8a, b, c, d would always remain active waiting for jobs to process. A thread is an independent flow of control that operates within the same address space as other independent flows of controls. Multiple threads may be grouped into a single entity called a process. In a multiprocessor system, multiple threads can execute at the same time, one on each processor. There may be multiple processes executing, each managing the operation of multiple threads. A process, such as a worker 8a, b, c, d process, provides a common address space and common system resources, including file descriptors, signal actions, shared libraries, and inter-process communication tools (such as message queues, pipes, semaphores, or shared memory). A thread is the schedulable entity of execution activated within a process. Thus, the worker processes 8a, b, c, d may execute a thread to perform the worker program functions to process a job and set the next status level to trigger the next worker in the workflow.

Each worker 8a, b, c, d is associated with one input status and one or more output statuses. One worker 8a, b, c, d is designated to process those jobs having the input status associated with that worker. For instance, if the status of the job is print, then the print worker having an input status of print would handle jobs having the

print status. After completing processing the job, depending on the outcome, the worker 8a, b, c, d will set the job status to one of its output statuses.

Connections 16a, b, c, d between the workers 8a, b, c, d and the database 6 provide an interface between the worker 8a, b, c, d processes and the database 6 to allow communication there between in a manner known in the art. Communication can be established using database communication protocols such as Open Database Connectivity (ODBC). The user defined function (UDF) 12 may communicate with the workers 8a, b, c, d using an application communication protocol 18a, b, c, d known in the art such as Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), and Remote Procedure Call (RPC).

Through the database connections 16a, b, c, d, the workers 8a, b, c, d, may access the job status table 10 in the database 6. In preferred embodiments, the workers 8a, b, c, d are continuously connected to the database 6 through connections 16a, b, c, d.

In preferred embodiments, the user defined function (UDF) 12 is a program called from within the database program 6 to communicate with and instruct the workers 8a, b, c, d to notify the workers of a status change with respect to at least one job in the job status table 10. Preferred embodiments utilize a database trigger 14 to notify the UDF 12. In the IBM DB2 system, an after trigger is an object in the database 6 that is invoked indirectly by the database manager when a particular SQL statement is completed. In preferred embodiments, an instance of an SQL statement updating or inserting the status to the job status table 10 would invoke a trigger event to activate the trigger 14 that would, in turn, execute the user defined function (UDF) 12. In preferred embodiments, the trigger 14 would call the user defined function 12 with the new status entered into the status column.

The user defined function 12 would include a mapping that would map an input status of a job to a worker 8a, b, c, d designated to process jobs having the particular input status. This mapping would allow the user defined function 12 to

send a message to the worker 8a, b, c, or d designated to process the current status. The user defined function 12 would send the worker 8a, b, c, information such as the job status.

FIG. 3 illustrates logic implemented in the database 6 and user defined function (UDF) 12 to process changes to the job status table 10. With respect to the database 6, control begins at block 100 where the database 6 receives an update to the status column of a job. This update could be setting the job to the initial status of ready, or any other possible status value. This updating or setting of the status in the job status table 10 is a triggering event that activates the job status trigger (at block 102). The activated job status trigger then executes (at block 104) the UDF 12 and passes to the UDF 12 the new status for the job. In alternative embodiments, the job status trigger could pass the UDF additional information such as the job ID of the job having its status changed.

With respect to the UDF 12, at block 120 in FIG. 3, the UDF 12 is called by the job status trigger, which passes the new job status. The UDF 12 then processes a mapping (at block 122) encoded in the UDF 12 program to determine the worker process 8a, b, c, d designated to handle the status passed from the database 6. The UDF 12 program then calls or sends a message to the worker process 8a, b, c or d whose input status is the new job status. Calling the worker 8a, b, c or d would cause the worker to query the database for any jobs having the input status for that worker if the worker 8a, b, c, d was not already actively processing the job status table 12. Alternatively, the UDF 12 could further pass the job ID, and then the worker 8a, b, c, d would access the specific job in the job status table 10, instead of querying the job status table 12 for the highest priority job having the input status.

When a job enters the workflow environment, its status would be set to ready to cause the worker that performs the first operations in the workflow process to process the new job. Each worker 8a, b, c, d processes jobs in a similar manner, although the actual operations will vary depending on the specific operations

performed by the worker 8a, b, c, d in the workflow process. To define a worker 8a, b, c, d, the conditions must be specified in the database for the trigger that notifies the UDF 12 to notify the appropriate worker. Each job may have an associated priority, shown in the priority column of the job status table 12 in FIG. 2. In such case, the worker 8a, b, c, d would query the job status table 12 for the highest priority job having the worker's input status. If a time stamp is provided indicating when the job was added to the table 12, then the worker 8a, b, c, d could query the table 12 first on priority, then on the oldest time stamp, or longest pending job.

As discussed, after processing a job, the worker changes the job status to one of the worker's output statuses. In all cases, the worker 8a, b, c, d must change the status for the job in the job status table 10, otherwise the worker 8a, b, c, d will continue processing the job if its status remains unchanged.

Upon system initialization, the workers 8a, b , c, d would be activated at the same time to allow a job to flow through and be processed by all workers. Any workers 8a, b, c, d that are not activated could form a bottleneck in the workflow system. As discussed, in preferred embodiments, the workers 8a, b, c, d are daemon processes executing in one or more systems that remain active at all times. The activated worker 8a, b, c, d establishes a connection 16a, b, c, d with the database 6.

FIG. 3 illustrates logic implemented in a worker process to participate in the workflow management system. Control begins at block 150 with the worker 8a, b, c, d waiting for a call from the UDF 12. At this state, the worker is sleeping and not processing the job status table 12. In response to receiving a signal from the UDF 12, the worker 8a, b, c, d queries (at block 154) the job status table 12 for a job having the input status for the worker. The worker 8a, b, c, d would ignore a call from the UDF 12 received when the worker 8a, b, c, d is actively querying and processing the job status table 12. The worker 8a, b, c, d then determines (at block 156) whether there are any unlocked jobs in the job status table 12 having the input status. If not, then

the worker 8a, b, c, d returns to block 150 where it enters sleep mode waiting for the next signal from the UDF 12.

If there are unlocked jobs having the input status, then the worker 8a, b, c, d accesses (at block 160) the highest priority job having the input status, marks (at block 164) the job as locked, and spawns (at block 166) a new thread to process the job. The worker 8a, b, c, d would pass the thread the database connection handle to allow the thread to access the database 6 through the connection 16a, b, c, d of the worker process 8a, b, c, d. The connection handle is a data object that contains information associated with a database connection, including general status information, transaction status, and diagnostic information. As discussed, if there are multiple jobs having the worker status, then the worker process 8a, b, c, d can spawn multiple, concurrently executing threads to process multiple jobs during the workflow process. From block 166 the worker 8a, b, c, d would return to block 156 to process any further unlocked jobs having the input status. If there are no more jobs with the input status, then the worker 8a, b, c, d returns to the sleep mode at block 150. With the logic of FIG. 4 the workers 8a, b, c, d continue querying the table for active jobs. If there are no further jobs in the job status table 10 having the status associated with the worker 8a, b ,c , d, then the worker goes into sleep mode until receiving a signal from the UDF 12 program indicating the occurrence of an update to the input status associated with the worker 8a, b, c, d.

As discussed, in alternative embodiments, the UDF 12 can pass the worker 8a, b, c, d the job ID that had its status changed to the worker's input status. In such case, the worker 8a, b, c, d would query the job status table 12 for the specific job having the passed job ID and process that job alone. After processing that job, the worker 8a, b, c, d could go back to sleep waiting for a call for the next job. In still further embodiments, the UDF 12 could call all workers 8a, b, c, d to process the job status table after being executed by the job status trigger.

When processing the job status table 12, the worker process 8a, b, c, d could spawn a thread for each located unlocked job in the job status table 12 having the input status. To prevent a worker process 8 from spawning enough threads to substantially degrade system performance, there can be a limit on the maximum number of threads a worker 8a, b, c, d can spawn to limit the resources allocated to each worker 8a, b, c, d. In alternative embodiments, the worker process 8a, b, c, d could process the job itself and perform the operations described with respect to the thread. Alternatively, the worker process 8a, b, c, d could spawn multiple instances of itself to have multiple instances of the worker simultaneously querying and processing jobs in the job status table 12. In cases where multiple threads or processes are processing jobs, the locking feature is used to prevent to processes or threads from simultaneously accessing the same job.

FIG. 5 illustrates logic implemented in a thread spawned by a worker process 8a, b, c, d at block 166 in FIG. 4. At block 200, the thread is spawned. The worker process 8a, b, c, d passes the identifier of the job to process. The thread then processes (at block 202) the job according to the actions encoded in the worker 8a, b, c, d spawning the thread in the workflow management system. For instance, in large scale printing systems, after a job is specified in a database, a first worker could generate the document, a second worker could process the job to format for a particular output device, and a third worker could route the formatted job to the output device. The thread at block 202 would perform the operations designated to the worker spawning the thread. If (at block 204) the processing completed successfully, then the thread updates the status for the job in the job status table 10 with the output status for the worker. This output status is associated with the next worker in the workflow that will process the job next. In this way, the input and output statuses defined for the workers 8a, b, c, d defines the flow of jobs to and from the workers 8a, b, c, d as defined by the workflow management system.

If the job did not complete successfully, then the thread would update the status for the job (at block 208) to the error state, which would trigger a worker that handles the error state to process the job in such error state. After updating the status at block 206 or 208, the thread will mark (at block 210) the job as unlocked to allow the next worker in the workflow to access the job and terminate (at block 212) the thread.

Preferred embodiments provide a database event driven system to automatically route a job through a series of operations or workflow to process the job in a just-in-time manner. The job status table 10 could contain information on the job utilized by the workflow threads to identify the location of the job in the workflow environment. As discussed, setting the status to an error state would trigger the database to signal the UDF 12 program to signal the worker handling error states. If the error worker corrected the problem causing the error state, then the error worker could set the status to ready and resubmit the job to the beginning, or at any other point, in the workflow process. With preferred embodiments, there is no need to poll the job status table 10 at intervals as the database triggers cause the signaling of the workers to process a job, thus utilizing a "push" versus a "pull" approach to workflow management and processing. Further, with the preferred embodiments the threads do not need to establish separate connections with the database 6 and may just use the connection 16a, b, c, d established by the worker process 8a, b, c, d.

In preferred embodiments, the database 6 would maintain an additional job history table that has a field for the job id, the status, and a time stamp when the status was set. Every time the status for the job is changed, the new status is added to the job history table. This additional table may be used for tracking purposes to allow a tracking program to determine where a job has been and its current position in the workflow process. An additional database trigger could be used that executes whenever the job status is changed, along with the job status trigger, and writes

information to the job history table on the new status, including the job ID, new status, and a time when the new status was set.

With preferred embodiments, the workflow and order of job processing may readily be modified by modifying the mapping in the UDF 12 program to add, remove or modify the statuses associated with workers. This would alter the definition of which worker is invoked for a particular status. Further, preferred embodiments are completely event driven. When a worker is finished processing a job and changes the job status, then the job is passed to the next worker for this next state to process. No high level control program polling the job status table is needed.

The work processed by the workers could be any program or device that is modified according to computer processing. The work entity may be a computer readable file on which processing and routing operations are performed to process the data in the file and route the file to its ultimate, intended destination. Alternatively, the work entity may be a physical device that is processed through the workflow system. For instance, the job may be a semiconductor that flows through different work stations for sputtering and fabrication operations; the job may be a circuit board or any other device that is processed on an automated assembly line. Each worker 8a, b, c, d may manage the operations of each station in the automated assembly line. In such embodiments, the job status table indicates the status of the device being processed. The workers would process the device and route it to a next destination in response to signals from the event driven database. For instance, if the device is an electronic device, then the different workers may cause machine tools and other hardware to perform one or more operations on the device before having the next worker perform the next step of processing the electronic device. In such embodiments, the workflow system of the invention is used to rout a piece of work through an automated assembly system of robotic and non-robotic workers assembling and otherwise acting upon the work entity.

Alternative Embodiments and Conclusions

This concludes the description of the preferred embodiments of the invention. The following describes some alternative embodiments for accomplishing the present invention.

The preferred embodiments may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The term "article of manufacture" (or alternatively, "computer program product") as used herein is intended to encompass one or more computer programs and data files accessible from one or more computer-readable devices, carriers, or media, such as a magnetic storage media, "floppy disk," CD-ROM, a file server providing access to the programs via a network transmission line, holographic unit, etc. Of course, those skilled in the art will recognize many modifications may be made to this configuration without departing from the scope of the present invention.

In preferred embodiments, the status was indicated in a relational database program that includes triggers that respond to updates to the status fields in a database table. In further embodiments, the status information and event driven triggers may be implemented in data structures other than database structures.

The workers 8a, b, c, d were describes as daemon processes that maintain a constant connection with the database. However, in alternative embodiments, the workers 8a, b, c, d may comprise application programs that are initialized in response to the trigger event and have to reestablish communication with the database upon each initialization.

In summary, preferred embodiments disclose a method, system, and program for processing a job in a workflow environment. A signal is generated when status for the job is changed from a first status to a second status. A work process associated with the second status is notified that one job had its status changed to the second status in response to the signal. The work process processes the job that had

its status changed from the first status to the second status and modifies the status of the job after completing the processing of the job.

The foregoing description of the preferred embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

**Microsoft, Windows, and Windows NT are registered trademarks and Microsoft SQL Server is a trademark of Microsoft Corporation; DB2, AIX, OS/390, OS/400 and OS/2 are registered trademarks of IBM, MVS is a trademark of IBM; and Oracle8 is a trademark of Oracle Corporation; JAVA is a trademark of Sun Microsystem, Inc.